Collaborative Linked Open Data Editor

Final Year Project

Ali Akram	**2008-NUST-BIT-159**
Muhammad Umar Farooq	**2008-NUST-BIT-123**

A project report submitted in partial fulfillment of the requirement for the degree of Bachelors in information technology

Department of Computing

School Of Electrical Engineering & Computer Sciences

National University Of Science & Technology

Islamabad, Pakistan

2012

PROJECT APPROVAL

It is certified that the contents and form of thesis entitled **“Web Based Linked Open Data Editor”** submitted by *Ali Akram* **(2008-NUST-BIT-159),** Muhammad Umer Farooq **(2008-NUST-BIT-123)** have been found satisfactory for the requirement of the degree.

Advisor: __________________________

(Dr. Khalid Latif)

Co-Advisor: __________________________

(Sana Khaliq)

We want to dedicate our work to our parents, who always prayed

for our success, encouraged us for -hard work and dedicated all their life

to make our life comfortable.

Surely they are the pillars of strength for us.

Table of Contents

List of Figures:

List of Abbreviations:

MVC	Model View Controller
CLODE	Collaborative Linked Open Data Editor
SOAP	Service Oriented Architecture Protocol
REST	Representational State Transfer
WSDL	Web Services Description Language
RDF	Resource Description Format
DAML	Darpa Agent Markup Language

Abstract

Ontology is widely used as framework to model the domain knowledge. Knowledge modeling for a specific domain is done by multiple agents. At a time only one agent works on the model and passes it to the next agent when his part is done. It makes a long time to develop a complete ontology.

A lot of tools have been already developed to design and maintain centralized ontology for various domains. All these tools provide schema management but lack in linked data management and real time collaborative knowledge modeling. So there is need of a tool that provides an environment where a group of knowledge experts can do collaborative knowledge modeling in real time.

The proposed system consumes the following objectives:

A group of people can work, collaborate and manage ontology in real time. The use of web socket enables the change to reflect on each user's screen instantly. Moreover, everyone is aware of up-to date model. Collaborative work reduces time cost. Web socket helps to resolve the inconsistency and conflicts among different agents instantly.

In a nut shell, this tool provides real time collaborative knowledge sharing and modeling along with the basic functionalities that previously developed tools provide e.g. Ontology creating and editing.

Chapter 1

INTRODUCTION

In context of knowledge sharing **Ontology** means a *specification of a conceptualization*. It is used to share common understanding of the structure of information in an organization. CLODE (Web Based Linked Open Data Editor) is a tool that will provide us a web based solution to create and manage Ontologies and share it between the people and software agents. It will also help people in an organization to administer Linked open data on a centralized repository collaboratively. It uses RESTful web services for communication that provide user friendly environment to its clients.

There are many tools available for ontology development and its management with their own consequences. But the functionality due to which CLODE dominates all previous tools is linked data management on central repository and designing ontology collaboratively. That means it provides collaborative interaction between users, consistency and structural integrity that make it stable.

Importance

Ontologies have become common on the World-Wide Web. In this era of Web Ontologies are being used on large scale ranging from large taxonomies categorizing Web sites (such as Yahoo!, Google) to the categorization of products for sale and their features(such as on Amazon).

Many disciplines now a day's develop standardized ontologies that domain experts can use to share and annotate domain knowledge in their fields. But there is no user friendly administration of Ontology where ontology is defined by a domain expert and others share it and interact collaboratively. Also most of the domain experts want linked data of their organization managed and manipulated at single place. So there is need for collaborative ontology development as well as administration of linked open data on a central repository.

Project Goal

The goal of the project is to:

- Develop a web based tool with support of collaborative ontology development.
- Provide a common place (i.e. central repository) for linked open data administration.
- Provide user friendly interaction with Ontologies and Linked open data.
- Improve performance by saving *time cost*.
- Reduce complexity and inconsistency in ontology management.

Ontologies

Ontologies are being used to share common understanding of the structure of the information among people and software agents. It is one common knowledge and data model that is computer readable and understandable. It facilitates domain experts to reuse domain knowledge. There is a concept of classes, relationship, instances and properties that make it easy to implement the domain concepts. It defines the real world association between objects and what the meaning of that association is. In specific term ontology development includes defining classes, creating relationships between them and applying different types of conditions and constraints on them.

RESTful Web Services

Representational State Transfer defines a set of rules to design Web Services and how resources are addressed and transferred using HTTP protocols. REST got populated in recent years and is being used and implemented on large scale organizations. It has such a large impact on Web that it dominated SOAP and WSDL based architecture designs because of its simplicity, ease in use, reliability and stability. In the beginning it didn’t have too much fame but with the passage of time major frameworks for REST have started to appear and are still being developed and become an integral part of JAVA 6.

Serialization

Serialization is the concept of flattening and marshalling data and objects. This concept is used when we have to transfer objects over the internet. Objects are flattened into a serialized form are send over the internet. SOAP and REST Web Services use serialization mechanism for communication. There are different formats are available for serialization For example: "**RDF/XML**", "**RDF/XML-ABBREV**", "**N-TRIPLE**", "**N3**" and "**TURTLE**“.

Chapter 2

LITERATURE REVIEW

Technology has started interacting more frequently than it has been doing. Design and development field has also emerged with the same trend. So, as a developer while designing and development there are number of factors to be considered i.e., ease of API use application scope and user interaction style etc. Ext JS is a good open source technology that provides rich UI widgets including windows, buttons, dropdown, dialogs and menu and user driver control approach. Ext JS gets its plus points due to less processing at the server side ultimately making application faster. Ext JS is compatible with any server processing POST requests and return data in structural format.

Background History

Ext JS was truly a set of utilities and extensions build as yahoo YUI framework by 2006, idea initiated by Jack Slocum. It started with version 1.1, initially named YUI-ext 1.1.With the time its use increases and it got much popularity in market. Company then officially released version 1.1.Far after integration of JQTouch and Raphael with Ext named the company as **sencha**.

Release 2.0

In December 2007, company released version 2.0.comprising more interactive features and comprehensive documentation and samples. For the time being backward compatibility to version 1.1 is not provided.

Release 3.0

July 2008, version 3.0 released with new emerging features and controls including list view and flash charting. Version 3.0 is having facility of backward compatibility to version 2.0.

Release 4.0

April 2011, company released version 4.0 having rich and enhanced elements and controls. Revised drawing and theme packages provided. One change in architecture was the use of MVC architecture that made the code easy and reliable.

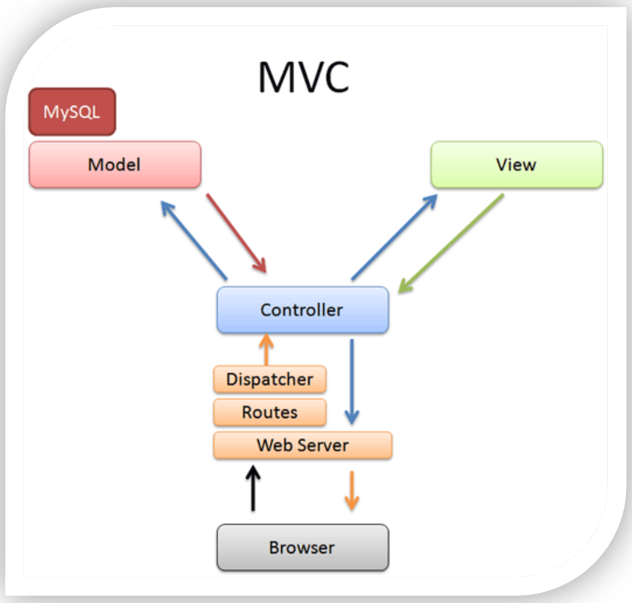
Application Architecture Overview

For application development to be successful and easy there is need of a well defined code organization and over all architecture. This helps to place it at best in market.

Architecture of this System is MVC architecture. That helps it organizing communication between clients and server in efficient way and writing organized understandable and reusable code.

MVC Architecture

It consists of three parts model, view and controller. There would be multiple views that will interact and interchange data with model and controller. A high level architecture diagram is shown below. Detailed architecture information would be provided later in next chapter.



(**Figure 2.1**)

Model

Model is invoked by controller. It communicates with underlying Data Store and ontology to fetch requested data and hands over this data to controller. And controller is responsible to generate response.

View

This layer is responsible for presentation of received data to the client side.

Controller

Controller receives requests from clients, fetches data from model and generates response accordingly. Requests could any of **PUT**, **POST** or **DELETE** data.

Comparative study of other Ontology Editors

Since with the change of data model and structure of ontologies, there are number of ontology editors in market observing different functionalities. But there are certain problems in the sense of interaction and collaboration while working on them. Before building up a new ontology editor there is need to have a look at the existing editors for their functionality, problems and scope. So a brief comparison of various features for major and prominent ontology editor is listed below:

Protégé

A web based as well as desktop tool using for browsing, editing and updating ontologies. It has the support for graph view. Have no support for collaborative work and exception handling. Furthermore, we cannot clearly express hierarchy if a class has more than one parent.XML and RDF is the supported serialization formats. Do not support for data administration.

OntoEdit

Normally support for the maintenance for ontologies through the graphical view. Serialization formats used are XML, RDF, DAML+OIL .Allow the users to have collaborative interaction to the ontology .Does not support for data administration.

WebOnto

A web based editor that supports collaborative browsing along with the editing and browsing the ontologies. Provide a graphical interface that presents the ontology as a mesh like structure. Have a limited support for checking consistency and data integrity. Do not support for exploring and data administration.

OilEd

It is a simpler tool for creating and analyzing ontologies on a limited scale. Do not support for XML format. Multiuser collaborative approach is also not available. Oiled is also lack of merging, exploring and data administration facilities.

After all, the proposed solution contains all the missing functionalities that the above ontology editors have. Moreover, there are two new features in this solution i.e. **Data administration** and **Classes Exploring** that are not in any ontology editor we have previously discussed.

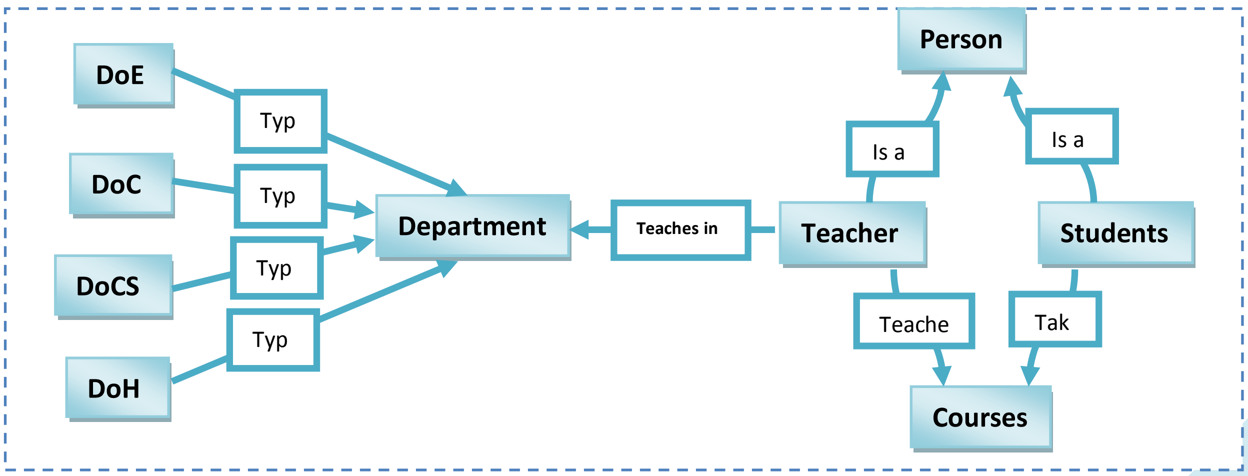
FUNCTIONALITY AND DESIGN

Web Based Linked Open Data Editor is a web based tool that will help domain experts to design ontology collaboratively that means users would be able to see instant changes in ontology as well as Linked Open Data administration. Let's demonstrate the functionality of this tool with an example.

SEECS Infrastructure Example

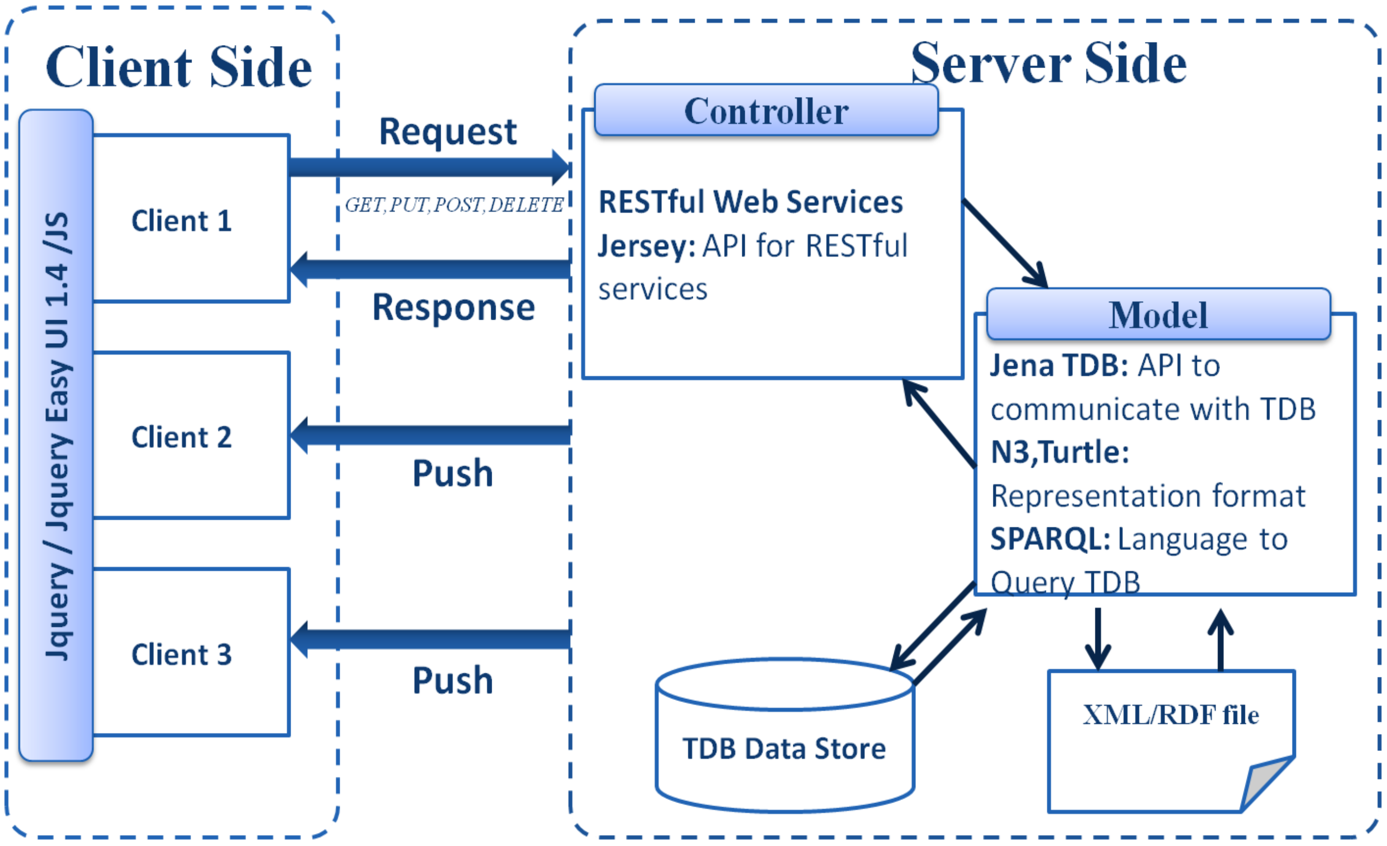
The basic idea is to provide a service that help to design the open infrastructure of SEECS. Dr. Arshad hired some professionals to design and manage SEECS infrastructure. They recommended him to use "Web Based Linked Open Data Editor" to design the domain model of SEECS. Dr. Arshad approved it. They designed the ontology collaboratively using this tool and shared it with each other (shown in figure 2). This ontology contains information of courses, departments, faculty, administration and students. There are 4 people each of them is managing data of one department (Dept. of Computing, Dept. of Electrical Engineering, Dept. of Basic Sciences and Dept. of Humanities). Now there is structural change request in infrastructure and there is a need to add a new faculty member, new student or some new department in the ontology. So administrator will just add information to the ontology and this change will reflect to the other members and they just don't need to update their own ontologies because they are working on the ontology that is on the central web server.

(Figure 3.1)



Application Architecture

Architecture followed to develop this web based application is MVC. It provided a lot of benefits in the design and development of the overall system. Architectural diagram of Web Based Linked Open Data Editor is given below.


Client Side
Jquery / Jquery Easy UI 1.4 /JS
Client 1
Client 2
Client 3
Request
GET, PUT, POST, DELETE
Response
Push
Push
Server Side
Controller
RESTful Web Services
Jersey: API for RESTful services
Model
Jena TDB: API to communicate with TDB
N3,Turtle: Representation format
SPARQL: Language to Query TDB
TDB Data Store
XML/RDF file

(**Figure 3.2**)

Architecture details

Client Side is implemented in JQuery easy UI/JavaScript and on the Server Side there is a controller that uses Jersey to implement RESTful web services and a Model that uses SPARQL to query the ontology and fetch results in JSON format. There is a sequence of events that occur when client requests some data from server.

Request

Client side generates request (GET, PUT and POST) to fetch data from server. Since multiple users are accessing and using ontology so, multiple requests may be generated from multiple clients.

Controller

On the server side Controller receives the request from client and on the basis of the argument passed it asks Model to fetch data from the Ontology file or Data Store. Controller implements RESTful web services using Jersey API. RESTful web service is used to represent resources with URI.

Model

After that Model quires TDB Data Store or Ontology file to fetch data. Query is written in SPARQL that is Query language of Semantic Web. Then Model returns results to the Controller.

Response

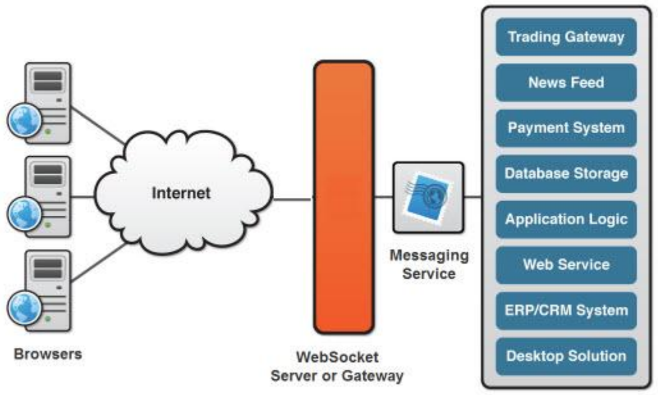
Now here comes the interesting part. Controller receives the data from Model and generates response to the client. In fact there are four kinds of request GET, PUT, POST and DELETE. In case of GET request it just returns the results to the client who requested the data. But in the case of PUT, POST and DELETE it affects changes on the server as well as there is need to tell other clients about changes affected in Ontology or Data store. To do this CLODE uses PUSH protocol to affect changes to the other clients and to implement PUST protocol it uses Web Sockets.

Traditionally a client server application uses half duplex architecture. Here in this case there are multiple users that are sending requests to the server at a time. And an updated view is required for each user, so pooling is one solution for this kind of situation.

But, the common issue with the client server application i.e. burden over the server and delayed response still lies there in this technique. Web socket help to avoid less traffic rendering among the client and server using PUSH based architecture.

Web Socket

Web Socket is client server technology used for real time full duplex communication. Mainly it is implemented in the web browsers. Before web sockets this type of communication was implementing using comet channels but they have overhead issues. Web socket solves these problems easily without comprising the security assumptions of the web.



(**Figure 3.3**)

How WebSocket Help in CLODE

As discussed above Web socket play an important role in CLODE. There are two kind of request that are generated whenever it needs changes in the model.

PULL

PUSH

PULL Request

Whenever a client needs data from web service it generates pull request that fetches data from the service and renders it on the client side using JQuery.

PUSH Request

This is the request that is generated from the server side whenever any kind of changes occur on the model server need to notify these changes to all the other clients so that they can render/ update their views also.

Chapter 4

RESULTS & DISCUSSION

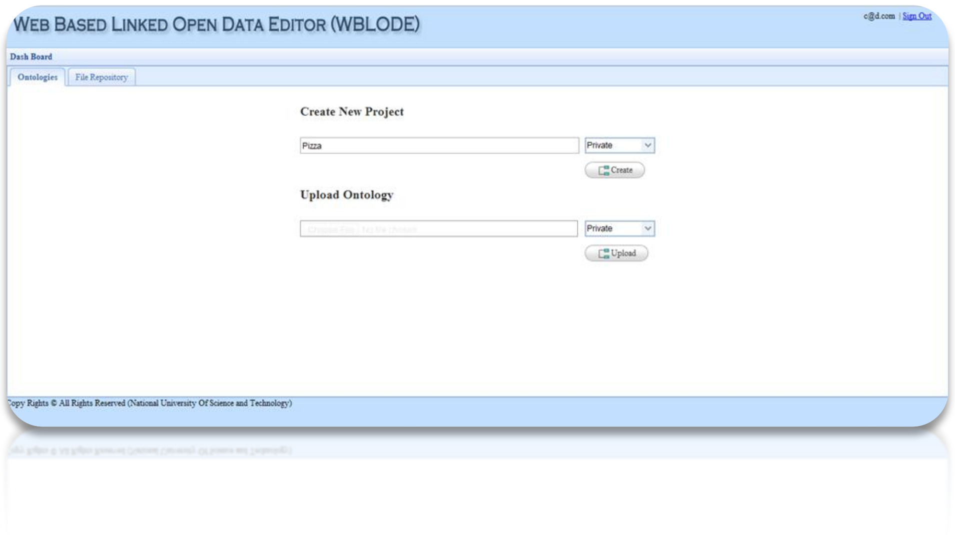
The implementation of the proposed solution lead to some useful results those not only include the basic principle of existing ontology editors rather some extended functionalities. The extracted results are mainly residing on both client and server end but on the basis of their functionality they are divided into following major components.

Basic Functionality

Like the other existing desktop and web based ontology editors, it facilitates the user to create, edit and update the ontology. Later we will discuss about them in detail.

Create ontology

User after having an account on CLODE can create a new ontology from scratch. User can create classes and properties and define their annotations accordingly class can have restriction on some property that will be editable and delete able. Below the figure shows the working environment while creating ontology.



Pizza

Private

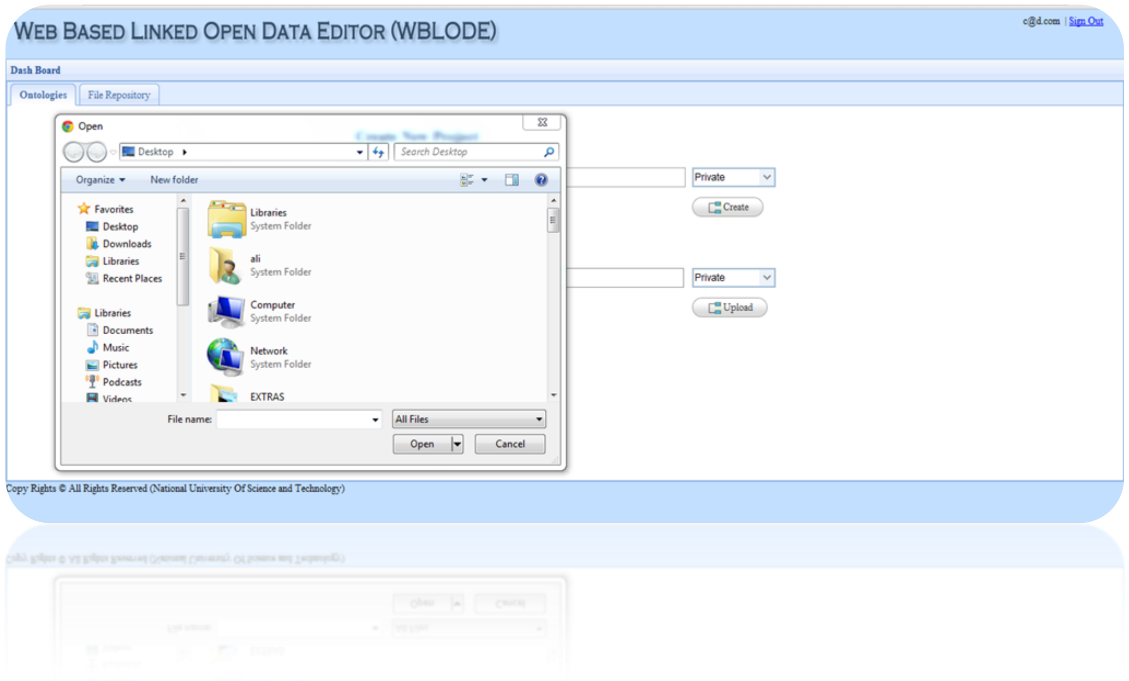
Private

Copy Rights © All Rights Reserved (National University Of Science and Technology)

(Figure 4.1)

Upload/Load from Repository

Once the user has created the ontology, he has options to share that ontology with the other users working on the same tool. There are access privileges are defined i.e. public and private. User is shown all the ontologies created by him, he can choose some user and name set some ontology to public or private for that user. This sharing mechanism helps to have access to their ontology and can manipulate that.



(Figure 4.2)

Collaborative Work

The second component and one of the major objectives of proposed solution is the collaborative work of people on CLODE at the same time. CLODE provides the functionality to multiple users to get login and work on certain ontology concurrently. This helps the concept to be built in relatively less time than the conventional solution takes. Changes and updating made by the other users are viewable.

Real time change reflection

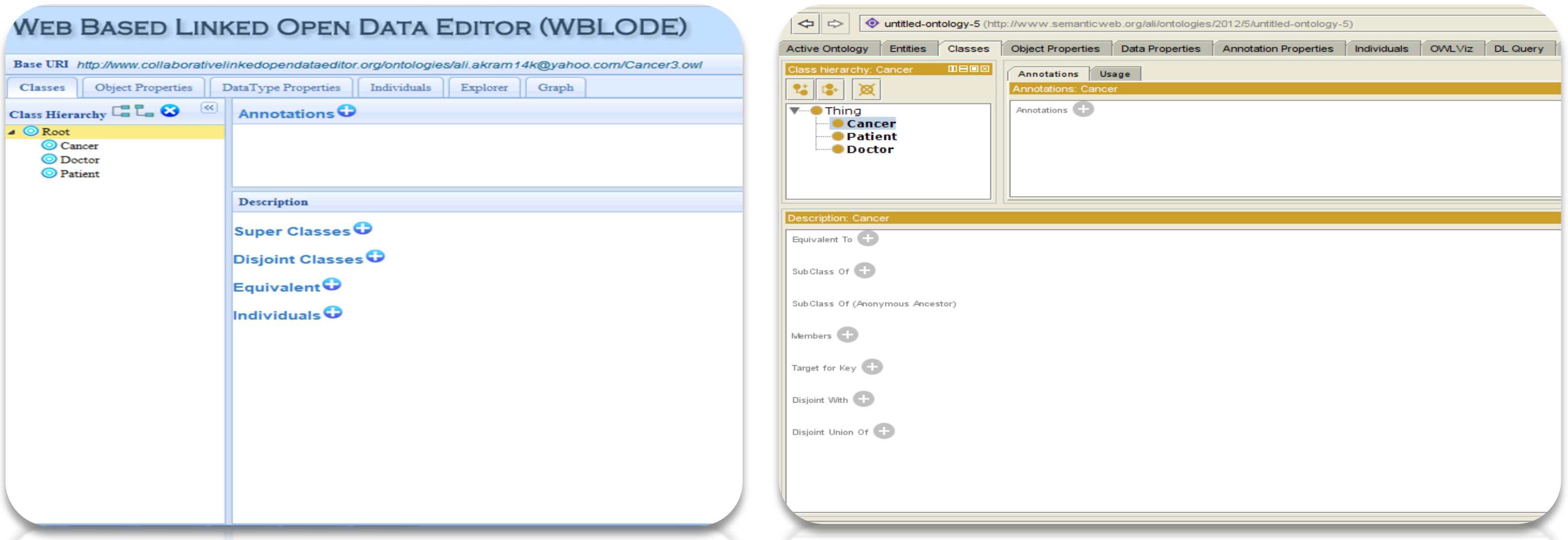
The most important and main objective achieved is the real time change reflection to other user for a particular change. This functionally not yet developed in any web or desktop ontology editor. So it provides the users more real time environment while developing some concept collaboratively.

Comparative Analysis

In order to test and verify the results of our tool we have created sample ontology of Cancer Diagnosis .There are two doctors as agents working on this ontology. After the making of ontology over the network by multiple clients working collaboratively we have tested the extracted results by making ontology on desktop protégé. Results are shown below.

Creating classes

As the class should be a noun so here the agent creates 3 classes named as Doctor, Cancer, and Patient. Below is the result of both CLODE and desktop protégé.

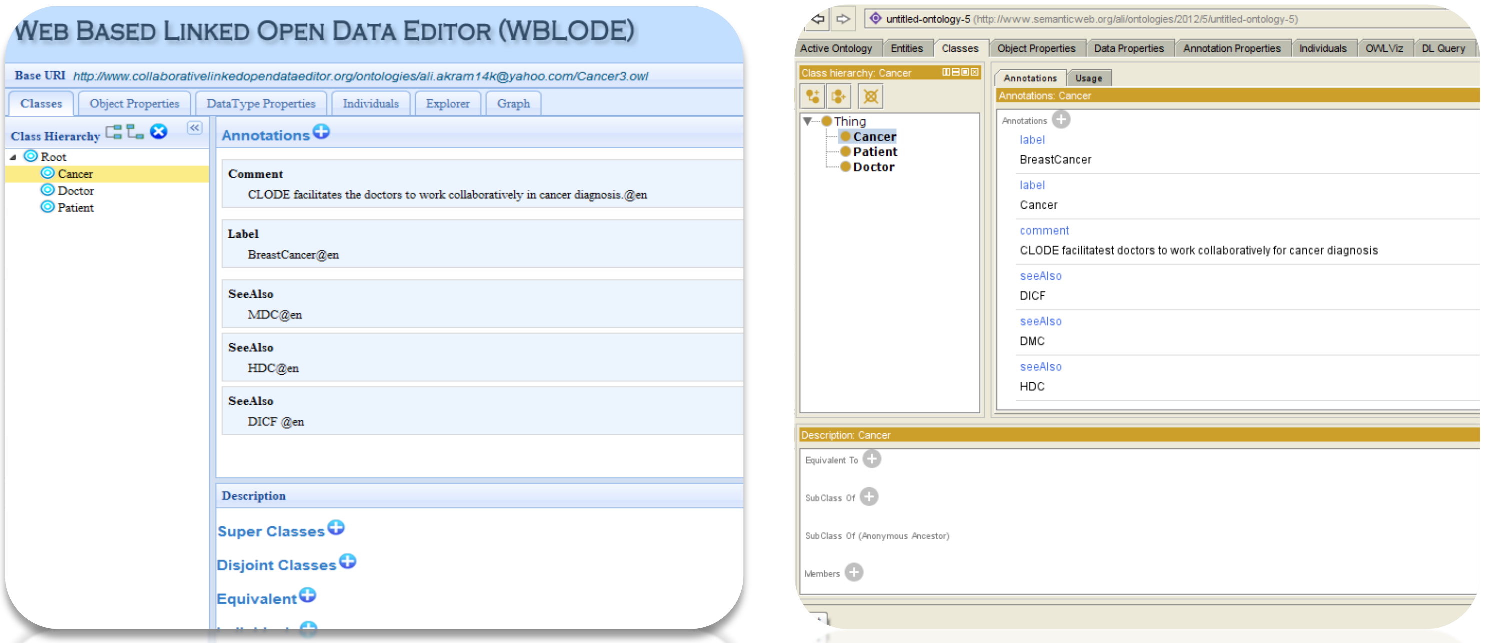


(**Figure 4.3**)

Adding Annotations:
After adding the classes the agents adds some annotations on these classes for the help of their identification easily.

- **Label**
- **SeeAlso**
- **IsdefinedBy**
- **VersionInfo**
- **Comments**

Agents can add the any of the annotations one or multiple times on a certain class. Below is the figure showing some added comments. Below is the result of both CLODE and desktop protégé.



Comment
CLODE facilitates the doctors to work collaboratively in cancer diagnosis.@en

Label
BreastCancer@en

SeeAlso
MDC@en

SeeAlso
HDC@en

SeeAlso
DICF @en

label
BreastCancer

label
Cancer

comment
CLODE facilitatest doctors to work collaboratively for cancer diagnosis

seeAlso
DICF

seeAlso
DMC

seeAlso
HDC

(Figure 4.4)

Adding Description:

Super Class

Here agent create a sample cancer class named as DMC and add its super as Cancer.

Restriction

Agent creates some restriction by selecting some property from object property tab and restricts it with some class selected from the classes tab.

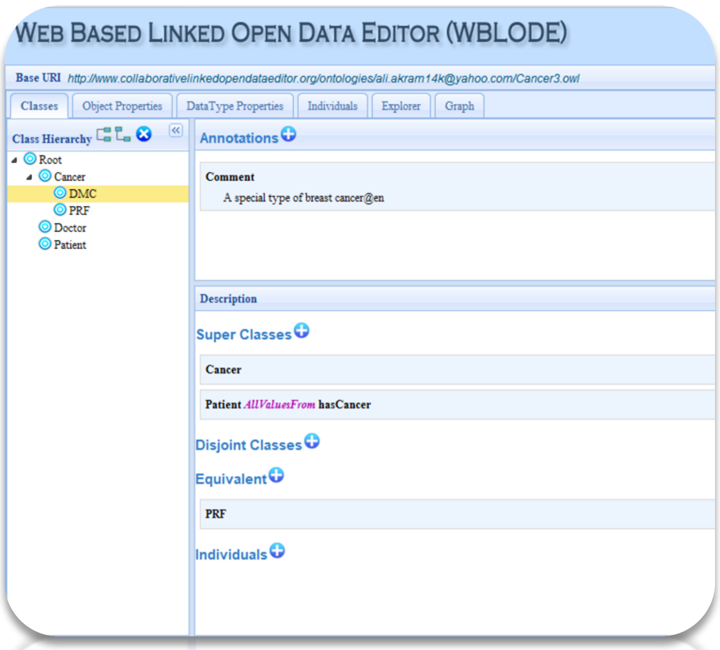
Individuals

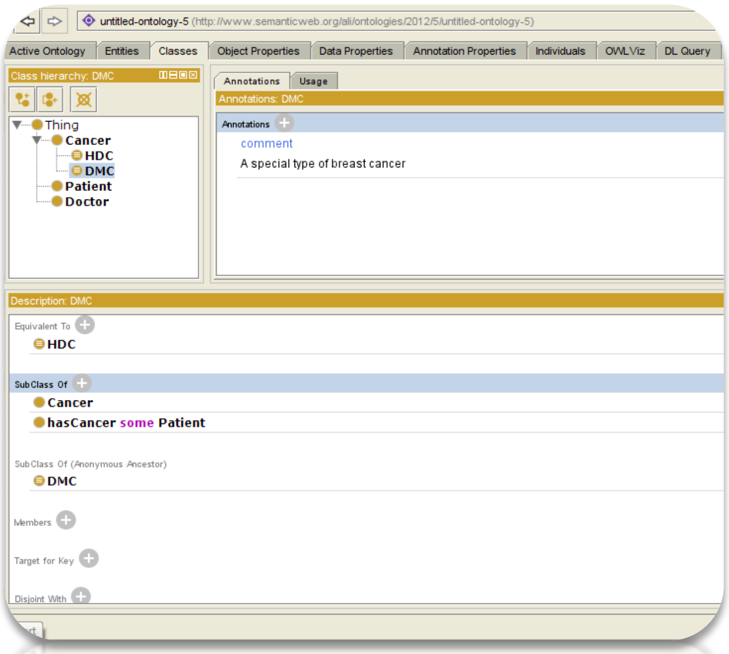
Agent adds some individuals for Doctor Class named as Smith, Jhon and Rony.

Equivalent
Agent class adds any class that has same properties as equivalent class to some other class.

Disjoint
Similarly can add disjoint class for some selected class.

Below is the result of both CLODE and desktop protégé.


WEB BASED LINKED OPEN DATA EDITOR (WBLODE)
Base URI http://www.collaborativelinkedopendataeditor.org/ontologies/ali.akram14k@yahoo.com/Cancer3.owl
Classes
Object Properties
DataType Properties
Individuals
Explorer
Graph
Class Hierarchy
Root
Cancer
DMC
PRF
Doctor
Patient
Annotations
Comment
A special type of breast cancer@en
Description
Super Classes
Cancer
Patient AllValuesFrom hasCancer
Disjoint Classes
Equivalent
PRF
Individuals



(Figure 4.5)

Comparative Analysis:

Below is a table showing a high level functionality comparison of our tool with the existing ontology editors.

Tool	Web Based	Multiple user access	Real-time Changes	Collaborative
Protégé	✗	✗	✗	✗
Web Protégé	✓	✗	✗	✗
OntEdit	✓	✗	✗	✗
CLODE	✓	✓	✓	✓

(Figure 4.6)

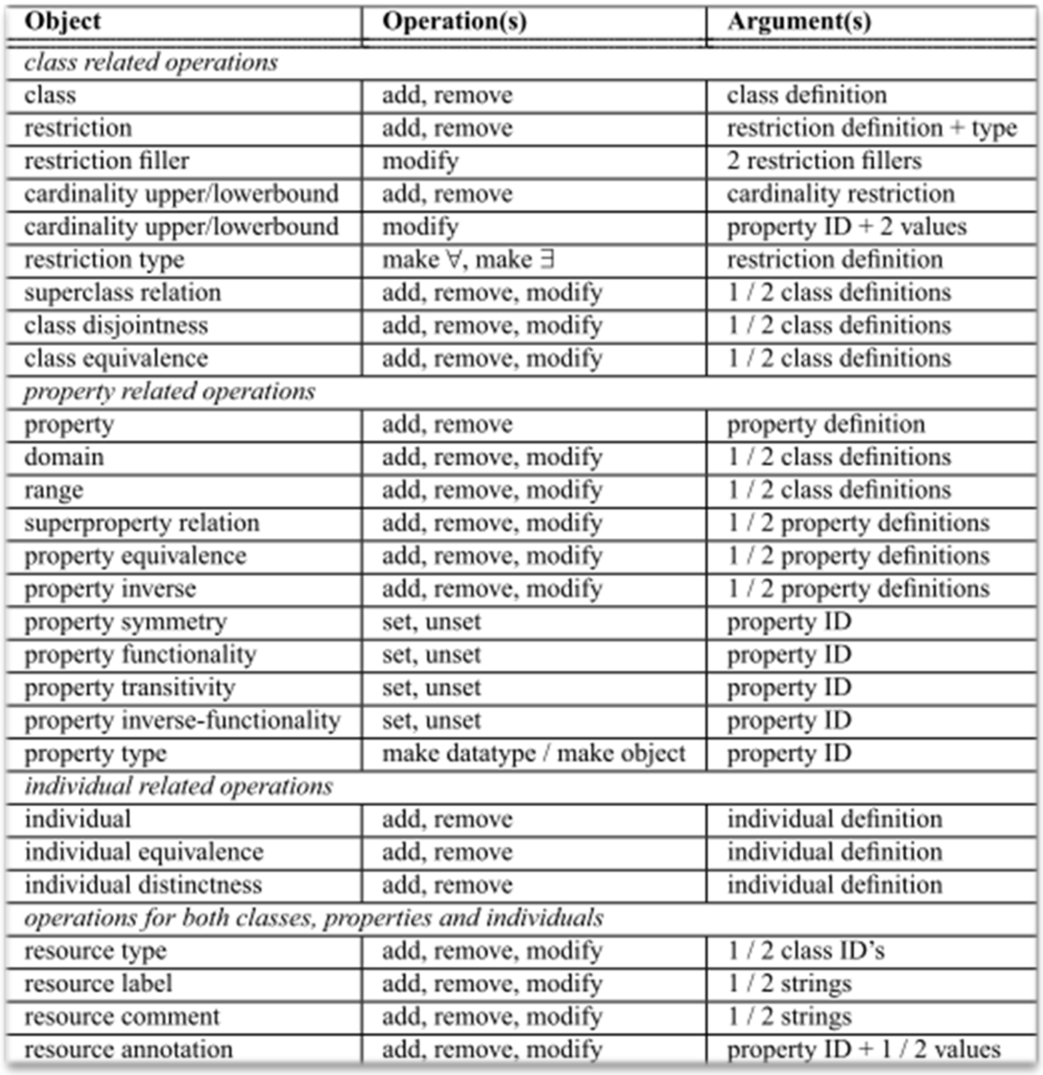
Discussion:

The web based linked open data editor is a complete package to create ontology and update/manipulate new as well as existing ontologies providing the extended functionalities and user friendly interaction.

Below is a snapshot of Michel Klein's PhD thesis written on topic **Change Management for Distributed Ontologies.** The research described in this thesis has been performed within the KIRMIT project and has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.

Here Michel has mentioned some must have change operations on ontology editing and according to him it is a challenging task to implement all these change operations in a single ontology editor.

(Figure 4.7)



Object	Operation(s)	Argument(s)
class related operations		
class	add, remove	class definition
restriction	add, remove	restriction definition + type
restriction filler	modify	2 restriction fillers
cardinality upper/lowerbound	add, remove	cardinality restriction
cardinality upper/lowerbound	modify	property ID + 2 values
restriction type	make ∀, make ∃	restriction definition
superclass relation	add, remove, modify	1 / 2 class definitions
class disjointness	add, remove, modify	1 / 2 class definitions
class equivalence	add, remove, modify	1 / 2 class definitions
property related operations		
property	add, remove	property definition
domain	add, remove, modify	1 / 2 class definitions
range	add, remove, modify	1 / 2 class definitions
superproperty relation	add, remove, modify	1 / 2 property definitions
property equivalence	add, remove, modify	1 / 2 property definitions
property inverse	add, remove, modify	1 / 2 property definitions
property symmetry	set, unset	property ID
property functionality	set, unset	property ID
property transitivity	set, unset	property ID
property inverse-functionality	set, unset	property ID
property type	make datatype / make object	property ID
individual related operations		
individual	add, remove	individual definition
individual equivalence	add, remove	individual definition
individual distinctness	add, remove	individual definition
operations for both classes, properties and individuals		
resource type	add, remove, modify	1 / 2 class ID's
resource label	add, remove, modify	1 / 2 strings
resource comment	add, remove, modify	1 / 2 strings
resource annotation	add, remove, modify	property ID + 1 / 2 values

Our ontology editor has not only implemented the changes operations mentioned by Klein rather it contains some extended functionalities as well.

Chapter 5

RECOMENDATIONS

Although Collaborative Linked Open Data Editor is providing all the required functionalities. It meets the objective of the project but we can improve this tool further by adding some more functionality in it. Some of recommend changes are given below.

- Access permissions
- Import/Export
- Link Multiple Ontology Models
- Visual Representation in the form of Graphs
- Developing ontology using Drag/Drop tool in Graphs
- Introduce in community
- Cloud Deployment

Access permissions

Currently there are two types of access permissions to the ontology files private/public. What we can do, we can put more restriction on the access for example before sharing model we can ask agent to specify the email addresses of the people with whom he wants to share the ontology. We can also set permission like if the model is readable only or writeable only few components.

Import/Export

Currently there is only feature of uploading the file to the ontology editor. Whenever user want to save his own file to the repository he needs to upload that file after login. We can further improve it by adding import/export feature in which a user can export the ontology file to his local system and do local changes and after that upload it again.

Link Multiple Ontology Models

There may be multiple ontology models that are being developed in an organization, maybe there is merger of two businesses and we need merge their models too. So we can add functionality of linking two models and resolving the conflicts between them.

Visual Representation in the form of Graphs

Currently in CLODE there are no graphs are being used for the visualizing model in the form of nodes and links between them. We can use some JavaScript based graph library like High Charts etc. to represent our ontology model in the form of graph, that would be helpful if want to have a look to our overall ontology and links between nodes.

Developing ontology using Drag/Drop tool in Graphs

We can also add a drag/drop feature for the adding/deleting of classes, properties and individuals and add links between them. In this way agents will be able to develop ontology in graphical mode using drag/drop feature.

Introduce in community

We can introduce CLODE in semantic web community so that people know its benefits and start using it frequently to get better performance.

Cloud Deployment

To make our solution faster and scalable we can deploy this product on the cloud, as we know the benefits of clouds like high performance, high availability data storage accessible via industry standard interfaces.

CONCLUSION

As we know that ontology are widely being developed for different domains because of its benefits like interoperability, reusability and fast searching, Collaborative Linked Open Data Editor will help organizations to model the concepts of their related domain collaboratively.

The organizations which are using Ontology in their businesses require changes and updating in their Ontology instantly. Before Collaborative Linked Open Data Editor there were a lot of tools being used for Ontology development but Collaborative Linked Open Data Editor provides a scalable solution for real time collaborative ontology development. Here are some non-functional features that CLODE is composed of.

- Real time collaboration
- Scalability
- User friendly interaction
- Central Repository
- Security
- Cost Effective
- Time Saving
- No Dependency
- Reduce inconsistency

Real time collaboration

In an organization multiple agents work in the form of groups to model the knowledge of some domain, so everyone should have knowledge of up-to-date model of that domain. CLODE provides functionality to see instant changes being done on the model instantly that will keep the Ontology model up-to-date.

Scalability

Scalability has been a big issue for the web based applications. When number of users increase web based applications dose not perform well. Usage of Restful web services and in memory processing of the models in CLODE provides scalable solution, in which increase in the user requests does not affect application performance too much.

User friendly interaction

Interactivity in the application makes it useable and use full. CLODE uses JQuery libraries that provide user friendly interaction as well as simplicity with beautiful controls and components that increase the worth of application.

Central Repository

Keeping information about model up-to-date when multiple agents are working on the same model is difficult when every agent has a separate copy of that model. They face a lot of difficulties while merging and updating their files to get final result. Model needs to be collected from each agent when model is complete. CLODE provides a central repository for the storage of files and it is shared between each member of the group working on the same ontology.

Security

Security is a big issue when we have to place files on the central repository. CLODE provides feature of access permissions like private and public. If you don’t want to share ontology with others you can keep it private and to share it with others change the status of file to public, it will be visible to every other members automatically.

Cost Effective

In web applications there are different kinds of cost involved for example network cost, I/O cost, when we need to fetch data from the file stored on the Disk or fetching data from databases. In the case of CLODE it does in memory processing on the model, it loads model into the memory and after handling the multiple requests if the model is dirty it writes back model to the Ontology file. There are different kind of services running at the backend that keep on checking the model if it is dirty and needs to update the file.

Time Saving

In previously developed tools only one agent is able to work on the model file at a time that takes a lot of time. But CLODE provided collaborative platform for Ontology development that saves the time of developers developing the model.

Dependency Reduction

While working in a group agent are some time dependent on the output of other agents to do their tasks, it creates a lot of dependency if number of agents are large. CLODE provides the feature of multiple accesses to the same model through which multiple agents can access model at a time and can do instant changes. In this way time is saved and efficiency increases.

Reduce inconsistency

Some times changes in a file by multiple agents create inconsistency, for example it may occur that at the same time two agent are trying to change in same thing. CLODE using web sockets provide the mechanism of queue processing for multiple requests.

REFERENCES:

http://roseindia.net [Ajax and JavaScript]

http://docs.sencha.com/ext-js/4-0/#/guide/application_architecture [MVC Architecture]

http://www.xml.com/2002/11/06/Ontology_Editor_Survey.html [Ontology Editor Survey Result]

http://docs.sencha.com/ext-js/4-0/#!/guide/getting_started [Getting Started]

http://en.wikipedia.org/wiki/Ext_JS [EXT JS]

http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData [Linking Open Data]

http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/Banff_96_final_2.html

[Toward Distributed System of Large Scale Ontologies]

http://www.ibm.com/developerworks/webservices/library/ws-restful/

[RESTful web services The Basic]

http://www.dl.acm.org/citation.cfm?id=1041421

http://www-ksl.stanford.edu/kst/what-is-an-ontology.html [What is ontology?]

http://www.uni-koblenz-landau.de/koblenz/fb4/institute/uebergreifend/er/IFI/AGStaab/Teaching/SS09/sw09/Ontology101.pdf

[Macieg Janik, Information system and semantic web, ontology design principle: 2]

http://www.sencha.com/deploy/dev/docs/

http://www.sencha.com/forum

http://www.sencha.com/blog

http://openjena.org/

http://protegewiki.stanford.edu/

http://www.oracle.com/technetwork/articles/javase/index-137171.html http://www.ibm.com/developerworks/webservices/library/wa-jaxrs/index.html

http://www.w3.org/TeamSubmission/turtle/

http://en.wikipedia.org/wiki/Ontology_(information_science)

http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

http://www.vogella.de/articles/REST/article.html

http://www.ibm.com/developerworks/web/library/wa-aj-tomcat/